QoS and Traffic Engineering: MPLS, DiffServ and Constraint Based Routing

A project report done
in partial fulfillment of the requirements for the degree of
Bachelor of Technology

Submitted by
Amardeep Singh (981101) **Gagan Mittal (981105)**

under the guidance of
Dr. S. K. Nandi

Department of Computer Science & Engineering
Indian Institute of Technology
Guwahati

May, 2000

Abstract

Providing Quality of Service (QoS) and Traffic Engineering capabilities in the Internet is very essential to support the requirements of current and future Internet services. MPLS and Diffserv show a lot of promise for making this a reality. Unfortunately, current network simulators lack a robust and coherent implementation of these technologies. The aim of this project is to deliver a complete and integrated simulation environment for ns-2 network simulator to facilitate research in next generation QoS and Traffic Engineering technologies, and to provide new views on the same.

Contents

1 Introduction

In this era of Internet, IP or the Internet Protocol plays a very important role. It has enabled a global network between an endless variety of systems and transmission media. There's no sign that the phenomenal growth of the Internet will subside any time soon. One reason for IP's tremendous success is its simplicity. The fundamental design principle for IP was derived from the *end-to-end argument*, which puts *smarts* in the ends of the network– the source and destination network hosts–leaving the network *core* dumb. IP routers at intersections throughout the network need do little more than check the destination IP address against a forwarding table to determine the next hop for an IP datagram. If the queue for the next hop is long, the datagram may be delayed. If the queue is full or unavailable, an IP router is allowed to drop a datagram. The result is that IP provides a *best effort* service that is subject to unpredictable delays and data loss.

This limitation has not been a problem for traditional Internet applications like web, email, file transfer, and the like. But the new breed of applications, including audio and video streaming, demand high data throughput capacity (bandwidth) and have low-latency requirements when used in two-way communications (i.e. conferencing and telephony). Public and private IP Networks are also being used increasingly for delivery of mission-critical information that cannot tolerate unpredictable losses.

Thus we need a mean to provide consistent, predictable data delivery service. This is referred to as Quality of Service or QoS. QoS is the ability of a network element (e.g. an application, host or router) to have some level of assurance that its traffic and service requirements can be satisfied.

It becomes increasingly important to manage network effectively and utilize network resources efficiently to fulfill the requirements of various Internet services. But at present, it is managed rather inefficiently. For example, due to the topology driven nature of current Internet routing protocols such as OSPF etc. , a high load condition in the network almost always result in a condition where some links are highly congested and others remain under-utilized, resulting in inferior quality of service.

Traffic Engineering for the Internet is therefore one of the primary concerns today. Internet traffic engineering is defined as that aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks [ACE+]. Traffic Engineering encompasses the application of technology and scientific principles to the measurement, characterization, modeling, and control of Internet traffic. Traffic Engineering aims to facilitate efficient and reliable network operations while simultaneously optimizing network resource utilization and traffic performance. As it turns out, this is indispensable to provide Quality of Service, as it provides a means for network optimization and bandwidth provisioning.

As evident from the current directions, Multi Protocol Label Switching (MPLS) and Diffserv are going to provide next generation QoS and Traffic Engineering architecture for the Internet. There is a lot of development going on in the IETF working groups on Diffserv, MPLS, and Traffic Engineering, and elsewhere.

One of the problems for research in these topics is lack of a complete and coherent simulation environment for these architectures. We aim to correct this by providing an integrated simulation environment for MPLS, Diffserv, and Constraint-based Routing in ns-2 network simulator. Ns is very widely used by the networking community and is widely regarded as a critical component of research infrastructure. But it lacks simulation modules for few domains such as QoS Routing and support for others such as Diffserv and MPLS is incomplete and non-integrated.

Thus, we started **Qosrns** project with an aim of developing/enhancing ns simulation modules for Diffserv, MPLS, Constraint-based/QoS routing to provide an integrated platform for research into these areas.

As part of Qosrns project, a Qos Routing module based on [AWK+99] has been developed. Also, MPLS simulation environment mns has been extended to use qospf routing. This work is a part of ongoing research project IRoNet (http://www.tct.hut.fi/tutkimus/ironet/index.shtml) at Helsinki University of Technology in the field of *Quality of Service and Traffic Engineering for the Internet*.

This project is hosted at http://sourceforge.net/projects/qosrns. There exists a mailing list to discuss about Qosrns and also, code can be downloaded both as tarballs and from anonymous CVS.

In this report we discuss MPLS, Diffserv, and Constraint based routing to appreciate their role in providing Quality of Service and Traffic Engineering for the Internet. Then we discuss the qos routing implementaion for ns and its integration with MPLS simulation environment.

2 QoS Routing

Routing Protocols used in the current Internet are quite robust but they are focussed on "best effort" datagram delivery, and so pose quite a few limitations for new generation Internet services requiring better quality of service. In particular, they use "shortest path routing", optimized on one single metric, typically bandwidth or hop count. Also, alternate paths with acceptable but non-optimal cost can not be used to route traffic due to the "opportunistic" nature of these algorithms due to which they use the current shortest path for routing. This results in conditions where one path to a destination would be heavily congested and others under utilized.

QoS Routing is defined as a routing mechanism under which paths for

flows are determined based on some knowledge of resource availability in the network as well as the QoS requirement of flows [CNS98]. It aims to extend conventional routing in three major ways.

Firstly, multiple paths between two nodes have to be calculated which could satisfy different service requirements. For eg., there may be separate paths for different bandwidth ranges, and delay requirements. As an example, consider that there are two paths available between two nodes, one with a higher bandwidth but through a satellite link, and other with a lower bandwidth but on fiber link. First path could be used for video broadcasting and latency is not an issue though bandwidth is. The second may be more suitable for audio conferencing application.

Secondly, it has to take care that changes in network parameters do not result in frequent shifting of traffic trunks through different paths. This is especially important with QoS routing as there routing may be tied to frequently changing metrics like available bandwidth.

Thirdly, it should support alternate routing in case sufficient resources are not available in the shortest path.

Thus, under QoS Routing, paths for flows would be determined based on some knowledge of resource availability in the network, as well as the QoS requirement of flows.

QoS routing and resource reservation protocols complement each other. Resource reservation protocols such as RSVP [BZB+97] provide a method for requesting and reserving network resources, QoS routing allows the determination of a path that can accommodate the requested QoS. Thus, generally QoS routing is used with some form of resource reservation/allocation mechanism.

There are several issues related with QoS Routing, such as:

- How is QoS capability of each link determined, and resources reserved.
- What is the granularity of routing decision (source/destination address, or per flow).
- What are the routing metrics used (hop count, available bandwidth, delay).

One of the QoS routing protocols is described in [AWK+99]. It proposes extensions to OSPF to support QoS Routing. It proposes to add an extra field to Link State Advertisement (LSA) packets, which gives the available bandwidth on a link. It also proposes two algorithms to compute shortest path route to a destination with a bandwidth constraint. It also talks about various link state update policies to ensure that routes have up to date information about the network, at the same time taking care that router/network load is minimum. Qos routing module in ns implements this protocol for quality of service routing.

3 Multiprotocol Label Switching Architecture

In normal IP forwarding each router analyzes the packet's header, and each router runs a network layer routing algorithm. Packet headers contain considerably more information than is needed simply to choose the next hop. Choosing the next hop can therefore be thought of as the composition of two functions. The first function partitions the entire set of possible packets into a set of "Forwarding Equivalence Classes (FECs) which is a group of packets that require equivalent forwarding treatment across the same path. The second maps each FEC to a next hop.

MPLS [RVC01] is developed for reducing the complexity of forwarding in IP networks. It is particularly an approach for achieving the simplified connection-oriented forwarding characteristics of layer 2 switching technologies while retaining the equally desirable flexibility and scalability of layer 3 routing. MPLS introduces a new forwarding paradigm for IP networks in a very scalable and cost effective way.

3.1 MPLS Label

In MPLS, the assignment of a particular packet to a particular FEC is done just once, as the packet enters the MPLS domain. The FEC to which the packet is assigned is encoded as a short fixed length value known as a "label". Since a packet is assigned to a FEC when it enters the network, the ingress router may use, in determining the assignment, any information it has about the packet, even if that information cannot be gleaned from the network layer header. For example, packets arriving on different ports can be labelled differently. Different labels can be assigned depending upon from which router it entered the network. All this information is not available in conventional forwarding.

3.2 Routing by Label Switching

Once a label is assigned to a packet, no further analysis of the packet's header is needed, only label lookup is required. The considerations that determine how a packet is assigned to a FEC can become ever more and more complicated, without any impact at all on the routers that merely forward labeled packets. MPLS uses label-swapping forwarding paradigm known as label switching. A router that involves in label switching is referred to as label switching router (LSR). Label at each hop has only a local significance representing the next-hop for packet's belonging to each FEC.

The path along which MPLS packet traverses is called a label-switched path (LSP). At each hop across an LSP through an MPLS domain, the packet gets a new label value that determines an outbound interface to the

next hop, and its treatments. Since the mapping between labels is constant at each LSR, the LSP is actually determined by the initial label value at the ingress LSR.

3.3 Aggregation

The procedure of binding a single label to a union of FECs which is itself a FEC (within some domain), and of applying that label to all traffic in the union, is known as "aggregation". The MPLS architecture allows aggregation. We can control the granularity as we are free to aggregate such FECs to a single FEC or to a set of FECs or not aggregate them at all. Aggregation may reduce the number of labels needed to handle a particular set of packets, and may also reduce the amount of label distribution control traffic needed.

3.4 Label Stacks

In more general model, each packet carries not just one label but a label stack. An unlabeled packet can be thought of as a packet whose label stack is empty. The processing of a labeled packet is completely independent of the level of hierarchy. The processing is always based on the top label.

We can speak of the level m LSP for Packet P as the sequence of routers which begins with LSP Ingress that pushes on a level m label. Then all intermediate LSRs make their forwarding decision by label Switching on a level m label. It ends when LSP Egress makes a forwarding decision made by label switching on a level less than m.

This enables us to incorporate the features MPLS Hierarchy and LSP tunnels.

3.5 Explicit Routing

Sometimes it is desirable to force a packet to follow a particular route which is explicitly chosen rather than being chosen by the normal dynamic routing algorithm. This may be done as a matter of policy, or for traffic engineering. Conventional solutions include

- Source Routing which requires the packet to carry an encoding of its route along with it.
- Network Layer encapsulation which causes the packet to be tunneled.

It is possible to implement tunnel as a LSP and use label switching rather than network layer encapsulation. The set of packet which are to be sent through the LSP tunnel, constitute a FEC. To put a packet into an LSP tunnel, the transmit endpoint just pushes a label for the tunnel onto the label stack and sends the labeled packet to the next hop in the tunnel.

3.6 MPLS Hierarchy: LSP Tunnels within LSPs

Consider a LSP of MPLS domain-1. A pair of neighbors R1, R2 of this LSP may not be directly connected but may be connected through some other set of routers. If this set of routers also happens to be a part of MPLS domain-2, then R1 and R2 are end point of a LSP of domain-2. So when packet will travel through R1 a label will be pushed on Label Stack corresponding to LSP of domain-2 and all operation will take place on that label and it will be popped before leaving R2 so that routing at domain-1 is unaware of this intermediate domain. It is actually tunneling the packet within LSP and as that tunnel is a LSP tunnel, we achieve LSP tunnel within LSP.

The label stack mechanism allows LSP tunneling to nest to any depth and provide the possibility of hierarchical nature of MPLS domain.

3.7 Label Distribution Protocols

The MPLS Architecture defines a label distribution protocol as a set of procedures by which one LSR informs another of the meaning of labels used to forward traffic between and through them.

MPLS Architecture does not assume a single label distribution protocol. In fact, a number of different label distribution protocols are being standardized. Existing protocols have been extended so that label distribution can be piggybacked on them. New protocols have also been defined for the explicit purpose of distributing labels like LDP.

4 LDP

LDP[ADF+01] is one of the Label Distribution Protocol implemented as a set of procedures and messages by which LSRs establish LSPs through a network by mapping network-layer routing information directly to data-link layer switched paths.

LDP associates a set of FEC with each LSP it creates. The FEC associated with an LSP specifies which packets are mapped to that LSP. LSPs are extended through a network as each LSR splices incoming labels for a FEC to the outgoing label assigned to the next hop for the given FEC.

Label Distribution and Management

- Label Advertisement modes

  There are two Label Advertisement modes. LSRs exchange advertisement modes during initialization. Both modes can be used in the same network in the same time.

Downstream On Demand label distribution This allows an LSR to distribute a FEC label binding in response to an explicit request from another LSR.

Downstream Unsolicited label distribution LSR can distribute a FEC label binding without an explicit request.

- Label Distribution Control Modes

  The behavior of the initial setup of LSPs is determined by control mode. An LSR may support both types of control as a configurable option.

  Independent Label Distribution Control Each LSR may advertise label mappings to its neighbors at any time it desires.

  Ordered Label Distribution Control LSR may initiate the transmission of a label mapping only for a FEC for which it has a label mapping for the FEC next hop, or for which the LSR is the egress.

- Label Retention Mode

  There are conservative and liberal Label Retention Mode depending on whether an LSR maintains a label binding for a FEC learned from a neighbor that is not its next hop for the FEC.

LDP mechanism to enable the exchange of FEC/label mapping requires four categories of LDP Message Exchange.

Categories of LDP Message Exchange

- Discovery messages, used to announce the presence of LSR in a network.
- Session messages, used to establish, maintain, and terminate sessions between LDP peers.
- Advertisement messages, used to create, change, and delete label mappings for FECs.
- Notification messages, used to provide advisory information and to signal error information.

LDP mechanisms Discovery messages provide a mechanism whereby LSRs indicate their presence in a network by sending a Hello message periodically. When an LSR chooses to establish a session with another LSR learned via the Hello message, it uses the LDP initialization procedure which is used to negotiate parameters like LDP protocol version, label distribution method, timer values etc. Upon successful completion of the initialization procedure, the two LSRs are LDP peers, and may exchange advertisement messages.

Bi-directional nature of LDP enables single LDP session allows each peer to learn the others label mappings.

It also provides loop detection as a configurable option which provides a mechanism for finding looping LSPs in the presence of non-merge capable LSRs. The mechanism makes use of Path Vector and Hop Count.

Briefly, A LSP is set up as follows. The ingress LSR sends a Label Request message toward the egress LSR, which sends back a Label Mapping message back to the ingress LSR. During the propagation of these label messages, all LSRs on this path use these label information to set up their forwarding tables so that packets can be forwarded using the label headers. [LR]

5 MPLS and Traffic Engineering

The key performance objectives associated with traffic engineering can be classified as being either traffic oriented or resource oriented Traffic oriented performance objectives include the aspects that enhance the QoS of traffic streams like minimization of delay, maximization of throughput. Resource oriented performance objectives include the aspects pertaining to the optimization of resource utilization. Minimizing congestion by Load balancing is addressed through Traffic Engineering.

A popular approach to circumvent the inadequacies of current IGPs is through the use of an overlay model, such as IP over ATM. The overlay model extends the design space by enabling arbitrary virtual topologies to be provisioned atop the network's physical topology. The virtual topology is constructed from virtual circuits which appear as physical links to the IGP routing protocols.

MPLS is strategically significant for Traffic Engineering because it can potentially provide most of the functionality available from the overlay model, in an integrated manner, and at a lower cost than the currently competing alternatives. Equally importantly, MPLS offers the possibility to automate aspects of the Traffic Engineering function.[AMA+99]

Functional capabilities required to fully support Traffic Engineering over MPLS in large networks include the support for traffic trunk attributes, resource attributes and constraint-based routing framework.

5.1 Traffic Trunks and attributes

Traffic trunk is an aggregation of traffic flows of the same class which are placed inside a Label Switched Path. Essentially, a traffic trunk is an abstract representation of traffic to which specific characteristics can be associated. It is useful to view traffic trunks as objects that can be routed; that is, the path through which a traffic trunk traverses can be changed [LN+98]. A set

of attributes associated with traffic trunks which collectively specify their behavioral characteristics.

- Traffic Parameter Attributes such as peak rate, average rate, burst sizes etc.
- Generic Path selection and maintenance attributes. These include, for eg. path preference attributes in case of multiple administrative defined paths, adaptivity attribute which defines whether a route could be re-optimizes in case of changes in network state, and load distribution attributes.
- Priority attribute which defines the relative importance of traffic trunks.
- Preemption attribute which determines whether a traffic trunk can preempt another traffic trunk from a given path, and whether another traffic trunk can preempt a specific traffic trunk.
- Resilience attributes which determine what happens under a fault condition — whether the trunk is not rerouted, or rerouted only if sufficient resources exist in some other path, etc.
- Policing attributes determine the action to be taken when a traffic trunk becomes non-compliant. Possible actions could be to drop the packets, or pass them as best effort packets.

5.2 Resource Attributes

A set of attributes associated with resources which constrain the placement of traffic trunks through them. These can also be viewed as topology attribute constraints.

- Maximum Allocation Multiplier

  The maximum allocation multiplier (MAM) of a resource is an administratively configurable attribute which determines the proportion of the resource that is available for allocation to traffic trunks. This attribute is mostly applicable to link bandwidth. This could be setup, for eg., such that the resource is over allocated. This could be useful in cases when it is known that peak demand of traffic trunks do not coincide in time

- Resource Class Attribute

  It is used to group resources into sets. It can be used to implement many policies with regard to both traffic and resource oriented performance optimization. For eg., considering links as resources, even non-neighboring links could be added to a class and various policies could

be defined such as specific inclusion or exclusion policies for some specific traffic, or specifying relative preferences for traffic trunk placement among various sets of resources.

5.3 Constraint Based Routing

A "constraint-based routing" framework which is used to compute explicit paths for traffic trunks subject to constraints imposed by items 1 and 2 above and other topology constraints. The constraint-based routing framework does not have to be part of MPLS. However, the two need to be tightly integrated together.

So although generally referred as QoS routing, constraint based routing is really a superset of QoS routing, incorporating policy based routing and traffic engineering functionalities into QoS routing. Constraint based routing enables a demand driven, resource reservation aware, routing paradigm to co-exist with current topology driven hop by hop Internet interior gateway protocols [AMA+99].

In general, the constraint based routing problem is known to be intractable for most realistic constraints. However some simple heuristic algorithms could be used, though they may not always result in a mapping.

Two protocols are being standardized for constraint based routing in MPLS. One of them, CR-LDP [AM+01] is a modification of LDP to support constraint based routing. Other is RSVP-TE [A+01], a set of extensions to RSVP to support traffic engineering and MPLS label setup.

Attractiveness of MPLS for Traffic Engineering can be attributed to the following factors: (1) explicit label switched paths which are not constrained by the destination based forwarding paradigm can be easily created through manual administrative action or through automated action by the underlying protocols, (2) LSPs can potentially be efficiently maintained, (3) traffic trunks can be instantiated and mapped onto LSPs, (4) a set of attributes can be associated with traffic trunks which modulate their behavioral characteristics, (5) a set of attributes can be associated with resources which constrain the placement of LSPs and traffic trunks across them, (6) MPLS allows for both traffic aggregation and disaggregation whereas classical destination only based IP forwarding permits only aggregation, (7) it is relatively easy to integrate a "constraint-based routing" framework with MPLS, (8) a good implementation of MPLS can offer significantly lower overhead than competing alternatives for Traffic Engineering. [AMA+99]

6 Constraint Based LSP setup using LDP

CR-LDP is a set of procedures through which LSRs not just only exchange labels and set up LSPs but also incorporate Constraint based routing. It is an end-to-end setup mechanism of a constraint-based routed LSP (CR-LSP) initiated by the ingress LSR. It also include mechanisms to provide means for reservation of resources using LDP. [JA+02] It includes support for the following.

- *Strict and Loose Explicit Routing*

  An explicit route is represented in a Label Request Message as a list of nodes or a list of groups of nodes. When the CR-LSP is established, all or a subset of the nodes in a group may be traversed by the LSP. Such a group of node is called 'Abstract Node'.

- *Maintaining LSPID*

  LSPID is a unique identifier of a CR-LSP within an MPLS network. The LSPID is useful in network management, in CR-LSP repair, and in using an already established CR-LSP as a hop in an ER-TLV, i.e. The LSPID is used to identify the tunnel ingress point as the next hop in the ER. This ER-Hop allows for stacking new CR-LSPs within an already established CR-LSP.

- *Specification of Traffic Parameters*

  The traffic characteristics of a path are described in terms of a peak rate, committed rate, and service granularity. The peak and committed rates describe the bandwidth constraints of a path while the service granularity can be used to specify a constraint on the delay variation that the CR-LDP MPLS domain may introduce to a path's traffic. Along with this we have frequency and weight.

  The Frequency specifies at what granularity the CDR allocated to the CR-LSP is made available. The weight determines the CR-LSP's relative share of the possible excess bandwidth above its committed rate.

- *CR-LSP Preemption though setup/holding priorities*

  If a route with sufficient resources can not be found, existing paths may be rerouted to reallocate resources to the new path. This is the process of path preemption. Setup and holding priorities are used to rank existing paths (holding priority) and the new path (setup priority) to determine if the new path can preempt an existing path.

- *Route Pinning*

A CR-LSP may be setup using route pinning if it is undesirable to change the path used by an LSP even when a better next hop becomes available at some LSR along the loosely routed portion of the LSP.

- *Resource Class*

  The network operator may classify network resources in various ways. These classes are also known as "colors" or "administrative groups". When a CR-LSP is being established, it's necessary to indicate which resource classes the CR-LSP can draw from.

- *Handling Failures* through rerouting

CR-LDP Mechanism Information about all above capabilities are incorporated in LDP messages to make them CR-LDP messages. LSP is established in the same way as LDP. Constraints are handled as follows.

If an LSR receives label request messages and if LSR can support the CR-LSP Traffic Parameters then the LSR must reserve the corresponding resources for the CR-LSP. If, after possible Traffic Parameter negotiation, an LSR cannot support the CR-LSP Traffic Parameters then the LSR must send a Notification Message that contains the "Resource Unavailable" status code. A Label Request Message containing an explicit route information must determine the next hop.

An LSR should adjust the resources that it reserved for a CR-LSP when it receives a Label Mapping Message if the Traffic Parameters differ from those in the corresponding Label Request Message.

If an LSR receives a Notification Message for a CR-LSP, it should release any resources that it possibly had reserved for the CR-LSP.

7 Differentiated Services

Differentiated Services or diffserv [BBC+98] is a classification based mechanism to provide QoS for the Internet. Prior to DiffServ, Internet QoS efforts were focused on Integrated Services/RSVP framework. IntServ allows sources and receivers to exchange signaling messages which establish additional packet classification and forwarding state on each node along the path between them. In IntServ architecture, path setup and bandwidth reservation etc. takes place for each microflow. Thus the amount of state on each node scales in proportion to the number of concurrent reservations, and thus exhibits huge scalability problems.

DiffServ was designed to provide a simpler, more coarse approach to establishing differentiated classes of service for Internet data. DiffServ pushes

complexity to the edge of the network (edge routers) where traffic classification and conditioning would take place. Traffic classification would result in assigning of traffic to different Behavior aggregates (BA's). These BA's are identifies by a label known as DS Codepoint which is encoded with the data packet. Each DS Codepoint has an associated queuing/forwarding behavior called Per Hop Behavior or PHB. Thus, complexity of inner elements of network (core routers) is reduced significantly as there is no state associated with each microflow, and no traffic conditioning is required.

7.1 Differentiated Services Domain

A DS domain [Gro] consists of DS boundary nodes and DS interior nodes. DS boundary nodes interconnect the DS domain to other DS or non-DS-capable domains, while DS interior nodes only connect to other DS interior or boundary nodes within the same DS domain.

A DS boundary node applies traffic conditioning on the traffic according to the Service Level Agreements which may be in place between the service provider and network subscribers. For this purpose Traffic Conditioning Agreements or TCA's, which are a subset of an SLA, include parameters regarding traffic profiles, performance metrics (such as latency, throughput, and drop priorities), and instructions on how out-of-profile packets will be handled. Packet Classifiers at boundary nodes select packets based on various parameters such as source/destination addresses or ports, DS field, and other information. They are used to map incoming packets to a particular DS Codepoint.

7.2 Per-Hop Behavior

Per-Hop-Behavior or PHB is defined as the externally observable forwarding behavior applied at a DS-compliant node to a DS behavior aggregate. It is the job of the DS nodes to apply appropriate PHB to the packets. The PHB is the means by which a node allocates resources to behavior aggregates.

PHBs may be specified in terms of their resource (e.g., buffer, bandwidth) priority relative to other PHBs, or in terms of their relative observable traffic characteristics (e.g., delay, loss). These PHBs may be used as building blocks to allocate resources and should be specified as a group (PHB group) for consistency. PHB groups will usually share a common constraint applying to each PHB within the group, such as a packet scheduling or buffer management policy. The relationship between PHBs in a group may be in terms of absolute or relative priority (e.g., discard priority by means of deterministic or stochastic thresholds), but this is not required (e.g., N equal link shares). A single PHB defined in isolation is a special case of a PHB group.

DS Codepoint of a packet selects the PHB it receives at a DS node. Few standard PHB's have been defined which have standard codepoints but codepoints could also be locally configured.

PHB's are implemented in terms of buffer allocation, queuing policy etc. PHB specifications do not include the algorithm to use. In general, many implementation mechanisms could be suitable for a particular PHB.

In particular, two standard PHB's have been specified — EF (Expedited Forwarding) and AR (Assured Rate Forwarding).

The intent of the EF PHB [C^+] is to provide a building block for low loss, low delay, and low jitter services. The rate at which EF traffic is served at a given output interface should be at least the configured rate R, over a suitably defined interval, independent of the offered load of non-EF traffic to that interface. EF could be implemented using a priority queue, or a weighted round robin scheduler.

The AR PDB [SNH] is suitable for carrying traffic aggregates that require rate assurance but do not require delay and jitter bounds. This PDB ensures that traffic conforming to a committed information rate (CIR) will incur low drop probability.

AR provides packet delivery in four independent forwarded classes with three levels of drop precedence (green, yellow and red in increasing order). The policer causes those packets to which assurance applies to be marked green, and other as yellow or red. Red packers may be dropped at the egress node.

Deployment of the AR PDB requires that the network is well-provisioned enough so that the likelihood of green packets being dropped in case of congestion is very low.

Multilevel RED queue could be used to implement AR PHB at network nodes.

8 Differentiated Services with MPLS

DiffServ provides scalable edge-to-edge QoS, while MPLS performs traffic engineering to evenly distribute traffic load on available links and fast rerouting to route around node and link failures. Moreover, MPLS can be deployed over a wide variety of link layer technologies. The combination of DiffServ and MPLS [Fau01] presents a very attractive strategy to backbone network service providers with scalable QoS and traffic engineering capabilities using fast packet switching technologies.

There are two issues in providing MPLS support of DiffServ. First, the DS code point is carried in the IP header, but the LSRs only examine the label header. Secondly the DS code point has 6 bits but the EXP field (field in MPLS label header dedicated to selecting PHB) has only 3 bits.

There are two Ways to handle both issues *EXP-Inferred- PSC LSP (E-LSP)* or *Label-Only-Inferred-PSC LSP (L-LSP)*. LSR diffServ Label Switching can be discussed in four stages.

- *Incoming PHB determination*

  For E-LSP, the EXP-to-PHB mapping can be either preconfigured or explicitly signaled during the E-LSP establishment. The LSR determines the PHB to be applied to the incoming packet by looking up the EXP field in the EXP-to- PHB mapping. [Fau01]

  For L-LSP, the EXP-to-PHB mapping is a function of the PSC (PHB scheduling Class) carried on the L-LSP, and is set up during the L-LSP establishment. Therefore, the PSC (i.e. queuing and scheduling) is already known to the LSR based on the Label field. The LSR then determines the drop precedence, hence the PHB, to be applied to the incoming packet by looking up the EXP field in the EXP-to-PHB mapping [Fau01].

- *Outgoing PHB determination with Optional Traffic Conditioning*

  A DiffServ LSR may perform marking, policing, and shaping on the incoming traffic streams, potentially changing the outgoing PHBs associated with non-conforming packets in the incoming traffic streams [Fau01]. Thus, the incoming and outgoing PHB may be different.

- *Label forwarding*

  Each DiffServ LSR must know the DiffServ context for a label which consists of LSP type, Supported PHBs, EXP-to-PHB mapping for an incoming label and PHB-to-EXP mapping for an outgoing label

- *Encoding of DiffServ Information into Encapsulation Layer*

  The LSR determines the EXP value to be written to the outgoing packet by looking up the PHB in the PHB-to-EXP mapping [Fau01].

  The LSR determines the EXP value to be written to the outgoing packet by looking up the PHB in the PHB-to-EXP mapping [Fau01].

E-LSP determines the PHB of a packet solely from the EXP field, and thus can support up to only 8 PHBs per E-LSP. The EXP field conveys the queuing, scheduling, and drop precedence to the LSR. PHB signaling can be used to explicitly signal the supported PHBs during LSP setup, but is not required (i.e. preconfigured PHBs) [Fau01].

L-LSP determines the PHB of a packet from both the Label and EXP fields. The Label field determines the PSC (queuing and scheduling) while the EXP field determines the PHB (drop precedence). An arbitrarily large number of PHBs can be supported.

9 Qos Routing Implementation

Here, we discuss routing architecture of ns. Then we discuss implementation of Qos routing based on [AWK+99] as part of Qosrns project.

9.1 Packet Forwarding and Routing in Ns

Ns is a discrete event simulator targeted at networking research. Ns is an object oriented simulator, written in C++, with an OTcl interpreter as a frontend. The simulator supports a class hierarchy in C++ (also called the compiled hierarchy), and a similar class hierarchy within the OTcl interpreter (also called the interpreted hierarchy). There are various classes which deal with various aspects of the package, eg. Simulator, Node, Link, Queue.

Class Node A node in ns simulation is represented by an instance of `Class Node`. Node itself is an aggregate object consisting of a set of classifiers. It also contains

- an address or `id_`, monotonically increasing by 1 as new nodes are created.
- a list of neighbors
- a list of agents
- a list of routing modules

Class Classifier The function of a node when it receives a packet is to examine the packet's fields, usually its destination address, and on occasion, its source address. It should then map the values to an outgoing interface object that is the next downstream recipient of this packet.

In ns, this task is performed by a classifier object. Multiple classifier objects, each looking at a specific portion of the packet forward the packet through the node. A node in ns uses many different types of classifiers for different purposes.

A classifier provides a way to match a packet against some logical criteria and retrieve a reference to another simulation object based on the match results. Each classifier contains a table of simulation objects indexed by slot number. The job of a classifier is to determine the slot number associated with a received packet and forward that packet to the object referenced by that particular slot.

Figure 1 shows the structure of a default unicast node. Address classifier decides, on the basis of destination address, to which link should the packet be forwarded. If it is destined for the current node, it is forwarded to port

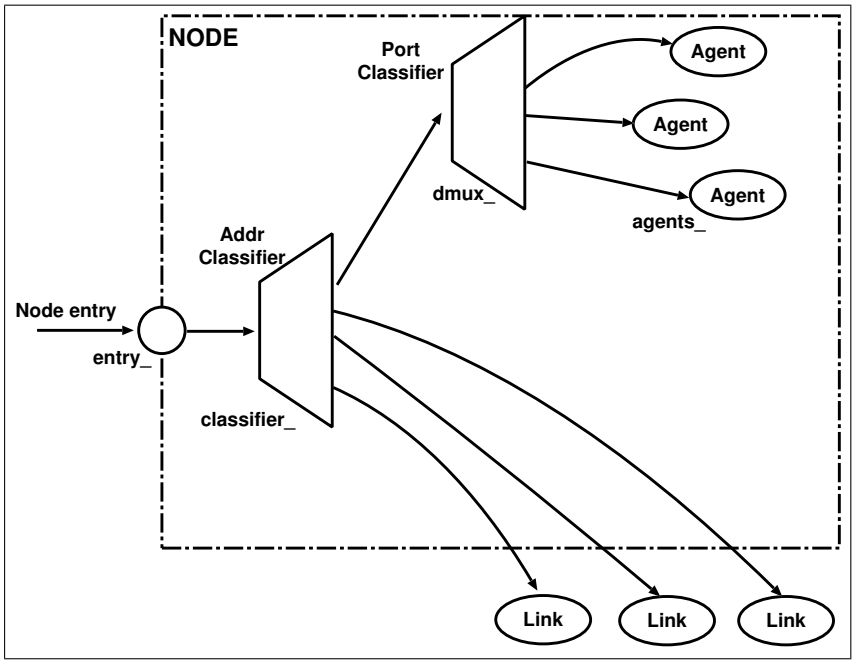

Figure 1: Structure of a Unicast Node

classifier which decides to which agent should the packet be forwarded based on port number.

Routing Module A ns node is essentially a collection of classifiers. The simplest node (unicast) contains only one address classifier and one port classifier, as shown in Figure 1. When one extends the functionality of the node, more classifiers are added into the base node, and each of these blocks requires its own classifiers. A provide a uniform interface to organize these classifiers and to bridge these classifiers to the route computation blocks is provided through the concept of routing modules.

In general, every routing implementation in ns consists of three function blocks:

- *Routing agent* exchanges routing packet with neighbors.
- *Route Logic* uses the information gathered by routing agents (or the global topology database in the case of static routing) to perform the actual route computation.
- *Classifiers* sit inside a Node. They use the computed routing table to perform packet forwarding.

A routing module manages all these function blocks and interfaces with node to organize its classifiers. Figure 2 shows the interaction between node, routing module and routing.

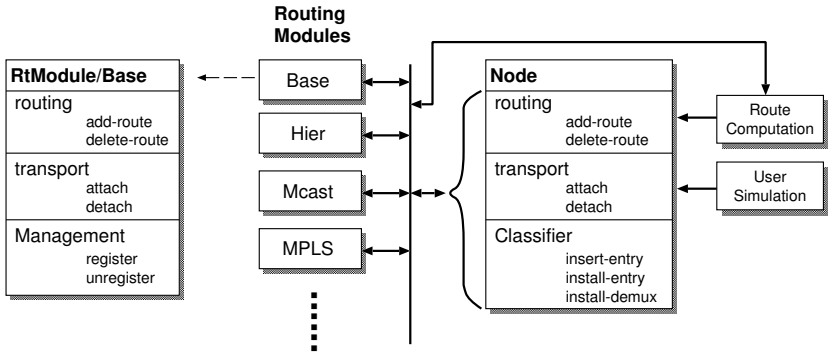

Figure 2: Interaction among node, routing module and routing

In order to know which module to register during creation, the Node class keeps a list of modules as a class variable. The default value of this list contains only the base routing module. The Node class provides the following two procs to manipulate this module list:

`Node::enable-module{[name]}` If module `RtModule/[name]` exists, this proc puts `[name]` into the module list.

`Node::disable-module{[name]}` If `[name]` is in the module list, remove it from the list.

When a node is created, it goes through the module list of the Node class, creates all modules included in the list, and register these modules at the node. After a node is created, one may use the following instprocs to list modules registered at the node, or to get a handle of a module with a particular name:

`Node::list-modules{}` Return a list of the handles of all registered modules.

`Node::get-module{[name]}` Return a handle of the registered module whose name matches the given one. Notice that any routing module can only have a single instance registered at any node.

Node provides the following instprocs to manipulate its address and port classifiers:

`Node::insert-entry{module, clsfr, hook}` inserts classifier `clsfr` into the entry point of the node. It also associates the new classifier with `module` so that if this classifier is removed later, `module` will be unregistered. If `hook` is specified as a number, the existing classifier will be inserted into slot `hook` of the new classifier. In this way, one may establish a chain of classifiers.

`Node::install-entry{module, clsfr, hook}` differs from `Node::insert-entry` in that it deletes the existing classifier at the node entry point, unregisters any associated routing module, and installs the new classifier at that point.

`Node::install-demux{demux, port}` places the given classifier `demux` as the default demultiplexer. If `port` is given, it plugs the existing demultiplexer into slot port of the new one.

9.2 Qospf Routing

This work implements a new routing module for ns. It consists of following components:

- A new routing logic that computes shortest path between two nodes with given available bandwidth as specified in [AWK+99],
- An agent — `Agent/QOSPF` that exchanges link state routing packets between various nodes, and
- Routing module `RtModule/QOSPF` which interfaces the above with `Node`

9.2.1 Ns Interface

`Node enable-module ''QOSPF'' :` Qospf routing module could be enabled in a simulation using `Node enable-module ''QOSPF''`. `RtModule/QOSPF` maintains a list of nodes on which "QOSPF" has been enabled. Later, this list is used to actually create instances of `Class Agent/QOSPF` on these nodes. To selectively enable "QOSPF" on few nodes, call `Node enable-module ''QOSPF''` before creating those nodes and call `Node disable-module ''QOSPF''` after creating those nodes.

`configure-qospf-nodes :` After all links have been created and queues initialized, `configure-qospf-nodes` should be called to create qospf routing agents on qospf nodes and initialize them. In case other routing modules eg. Mpls module have been enabled, they may install their own classifiers. Therefore it is important that `configure-qospf-nodes` is called after other routing modules have been initialized.

`SimpleLink instproc get-bw :` For calculating routes, qospf agent needs to know how much bandwidth is available. For this `SimpleLink instproc get-bw` needs to be implemented. It should return how much bandwidth is unreserved and available for reservation. In case the link does not participate in bandwidth reservation, it should return -1. Current available bandwidth could be found using various measurement modules or based on reserved bandwidth.

`RtModule/QOSPF bw-changed :` Whenever available bandwidth on a link changes, routing module should be notified using `bw-changed`. This would actually inform qospf agent running on that node about the bandwidth change, which would in turn update its tables. It would trigger linkstate updates and new route calculations.

`RtModule/QOSPF get-explicit route {dest, bw} :` This instproc in used to get the explicit route to destination `dest` along the shortest path with atleast `bw` available, as computed by qospf agent running on that node. There are two strategies of route computation — route precomputation and on demand route computation. Which of the two is used is determined by a configuration parameter as explained below.

`Agent/QOSPF onDemand :` This variable determines whether routes are precomputed or computed on demand. Actually, both the methods are always available, and this variable determines which of them is used by `get-explicit-root` function

`Agent/QOSPF advertInterval :` This variable determines periodic LSA update interval.

`Agent/QOSPF advertHoldDownInterval :` This variable determines the hold down interval for LSA updates (explained below in linkstate updates section).

`Agent/QOSPF advertMinBwChangePercent :` This variable sets the minimum percent by which available bandwidth on a link must have changed for link states updates to be sent (explained below in linkstate updates section).

Linkstate Updates An important consideration in qospf routing implementation is when should routing updates be send. As mentioned in [AWK+99], one option is to mandate periodic updates, where the period of updates is determined based on a tolerable corresponding load on the network and the routers. The main disadvantage of such an approach is that major changes in the bandwidth available on a link could remain unknown for a full period and, therefore, result in many incorrect routing decisions. Ideally, routers should have the most current view of the bandwidth available on all links in the network, so that they can make the most accurate decision of which path to select. Unfortunately, this then calls for very frequent updates, e.g., each time the available bandwidth of a link changes, which is neither scalable nor practical. In general, there is a trade-off between the protocol overhead of frequent updates and the accuracy of the network state information that the path selection algorithm depends on.

Therefore, a good strategy could be to send link state updates when available bandwidth on a link changes by a certain minimum percent. It could be coupled along with regular periodic updates. Also a periodic timer constraint in the form of a hold down timer can be applied so that link state updates are not sent too frequently.

Qosrns implementation of qos extensions to OSPF allows one to simulate different strategies by configuring in their simulation periodic update interval, hold down timer interval, and minimum bandwidth change percent by changing configuration variables `advertInterval`, `advertHoldDownInterval`, and `advertMinBwChangePercent` as explained above in section Ns interface.

10 MNS and QOS Routing Integration

MPLS Network Simulator is the module in ns-2 simulator which supports the establishment of CR-LSP and basic function of MPLS such as LDP and label switching.

10.1 Design of MNS

Conceptual Model and components of MNS

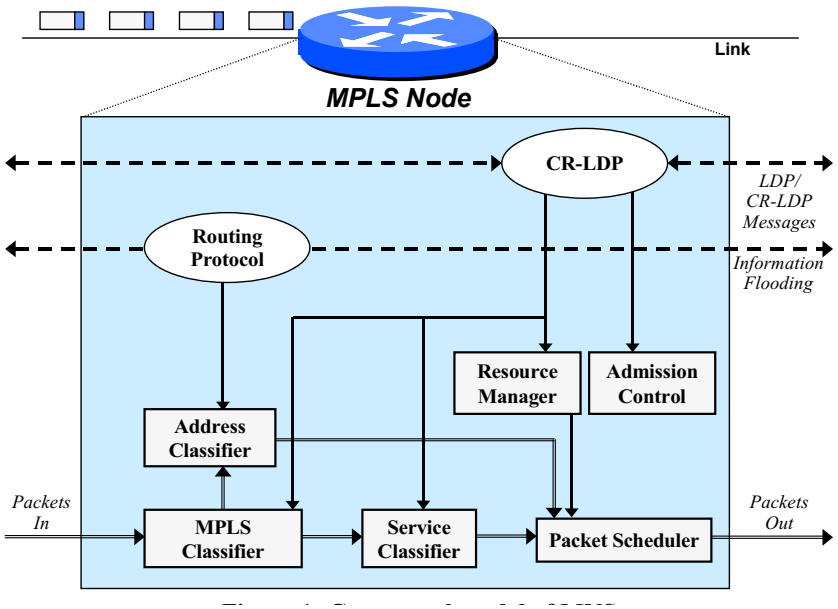

Figure 3: Conceptual Model of MNS

Components of MNS as shown in Figure 3 are following.

- *LDP/CR-LDP* - generates or processes LDP/CR-LDP messages

- *MPLS Classifier* - executes label operation such as push, pop and swap for MPLS packet.
- *Service Classifier* - classifier services that should be applied to the incoming packet by using label and interface information and associates each packet with the appropriate reservation
- *Admission Control* - looks at the Traffic Parameter of CR-LDP, and determines whether the MPLS node has sufficient available resource to supply the requested QoS
- *Resource Manager* - creates/deletes queues on-demand, and also manage the information of resource.
- *Packet Scheduler* - manages the packets in the queues so that they receive the service required.

Label Switching

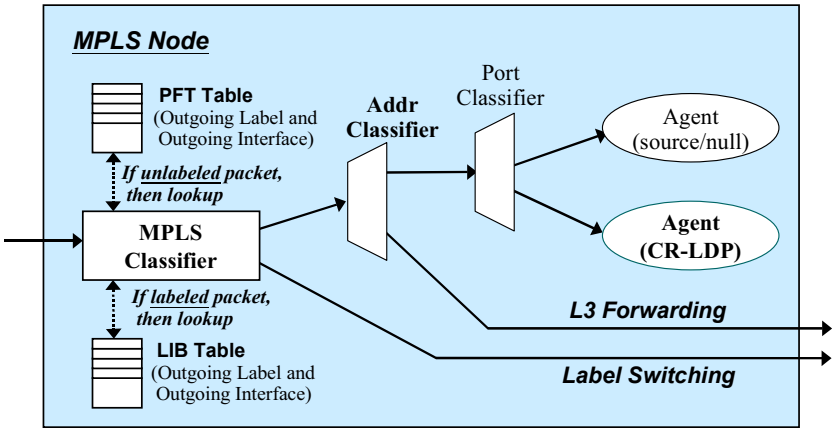

Figure 4: Architecture of MPLS node for Label Switching

MPLS classifier and LDP agent are inserted into IP node. As shown in Figure 4, when MPLS node receive a packet, MPLS classifier determines whether the received packet is labelled or unlabelled. If it is labelled, 'MPLS Classifier' executes label switching for the packet. If it is unlabelled but its LSP exists, it is handled like a labelled packet. Otherwise, it is send to 'Address Classifier' which executes L3 forwarding for the packets. Address Classifier executes L3 forwarding for the packets. If the next hop is the packet itself, the packet is sent to 'Port Classifier'.

For the label switching two tables are defined as follows;

- *Partial Forwarding Table (PFT)* - It is a subset of Forwarding Table and used to map IP packet into LSP at Ingress LSR. It consists of FEC, FlowID, and LIBptr. The LIBptr is a pointer that indicates a entry of the LIB table
- *Label Information Base (LIB)* - It has information for the established LSPs and used to provide label switching for the labeled packet. It consists of in/out-label and in/out-interface

MPLS QoS Traffic Processing

In order to support MPLS QoS traffic, the 'Service Classifier' component has been designed and implemented. CBQ, which has already implemented on NS, is selected for the 'Packet Scheduler' component.

A table is maintained called Explicit Route information Base (ERB). ERB contains LSPID and ServiceID for the established ER-LSPs.

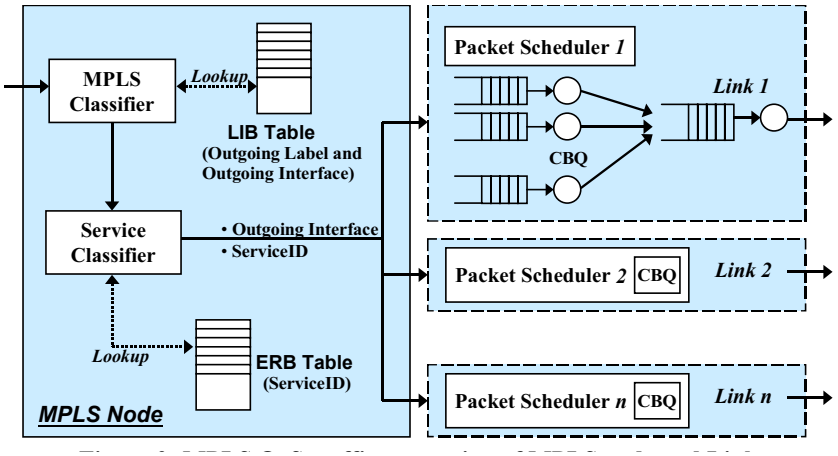

Figure 5: MPLS QoS traffic processing of MPLS node and link

As shown in Figure 5, MPLS node indexes the LIB table for outgoing label and outgoing interface to perform label operation and then the ERB table for serviceID to provide queuing service. According to the class the packet belongs to, it queues the corresponding buffer in CBQ. It is served by CBQ and transmitted to the outgoing-interface.

Resource Reservation

As in Figure 6, when 'CR-LDP' component receives a CR-LDP Request message, it call 'Admission Control' to check if the node has the requested resource. If there is a sufficient available resource, it is reserved and resource table is updated and the LDP request message is passed to the next MPLS node.

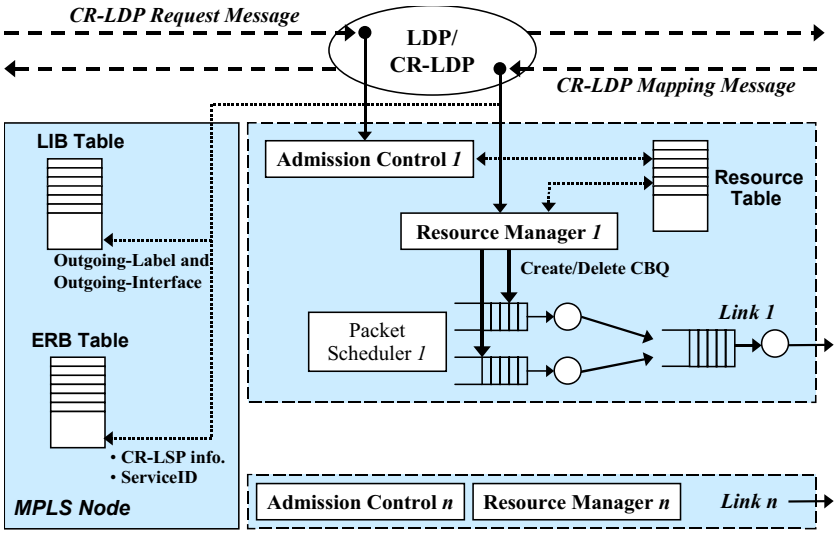

Figure 6: Resource reservation process of MPLS node and link

When 'CR-LDP' component receives a CR-LDP Mapping message, it saves the label and interface information in the LIB table and the requested CR-LSP information in the ERB table. Then it calls the 'Resource Manager' to create a queue to serve the requested CR-LSP, and saves its serviceID in ERB table. Then the mapping message is forward to next node.

10.2 Integration of MNS with QOSPF Routing Module

Integration means that if programmer while writing simulation script for MNS, enables QOSPF (while making MPLS nodes). Then, whenever a route computation is required, MNS will compute routes by OSPF algorithm with QoS extensions.

So if QOSPF is enabled then QOSPF agent also is attached to the node along with MPLS Classifier and CR-LDP agent. Briefly, following are the places where both the modules will need to communicate.

- *Initialization*

  Both modules will send their packets for getting information about links, bandwidths, topology map and will build their databases. QOSPF module should always take bandwidth information from MNS Resource Manager.

- *Route Selection*

  Whenever MNS module gets a request for a path having some minimum bandwidth then instead of using its own ways of route computation, the

parameters should be passed to QOSPF routing module, which should then compute the route and pass the route as a string of node ID to MNS and then reservation should be made along that route by MNS.

- *Route Formation*

  Whenever MNS node receives Label Request message and its Admission Control update the resource table, then QOSPF routing module should also update itself from there.

- *Route Failure*

  When the reservation of bandwidth for a route is in progress and if in between some node is found deficient of required bandwidth then notification messages for failure are send to previous nodes which then update their bandwidth databases accordingly. QOSPF module should update itself for each node here also.

- *Route Preemption*

  Sometime according to set/hold priority some CR-LSPs are preempted, at that time also bandwidth changes should be communicated.

10.2.1 NS Interface

`Simulator use-QOSPF-for-cbr :` It ensures that integrated module works and MNS use OSPF with QoS extensions for routing.

`Simulator configure-ldp-on-all-mpls-nodes :` It will make initialization in links, bandwidths and topology tables and will also make necessary initialization in QOSPF module

`Simulator configure-cbq-on-all-mpls-nodes :` It will install cbq classes on all MPLS node.

`RtModule/MPLS constraint-based-routing {dest bw} :` It will find a route with given bandwidth for the given destination using route computed by QOSPF agent.

`Simulator setup-crlsp {dest route lspID bw bSize pSize sPrio hPri} :` It makes the actual reservation and set up CR-LSP along the given route and make preemption and take care of route failures and simultaneously communicate the same to QOSPF module

11 Conclusion

Quality of Service and Traffic Engineering have taken a large prominence in recent times due to the explosive growth of the Internet and its commercialization. Although research in these fields has been going on for a long time, and has given rise to many solutions, their use in the Internet has been extremely restricted so far. This was partly due to previous solutions being too cumbersome to implement, and being insufficient in dealing with wide aspects of QoS and Traffic Engineering.

Multi Protocol Label Switching has emerged as an all encompassing solution which provides QoS and Traffic Engineering for the Internet. MPLS began as a low cost routing solution, but has gradually grown in scale to include other aspects like constraint based routing and traffic engineering. Yet, this is a very recent development, and hence a focus of lot of research, to augment its capabilities and integrate with other technologies like Differentiated Services and QoS Routing.

QoS Routing is an important aspect of Traffic Engineering, as it provides a mean to get paths based on available amount of resources such as bandwidth rather than using shortest path. Various ways to use QoS routing with existing/upcoming internet technologies have been proposed such as modifications to OSPF as in [AWK+99], and using LDP to interchange QoS routing information as in OSPF-TE [SJ02]

Qosrns project was started to implement QoS Routing in ns-2 network simulator. As part of this project, we have developed new modules in ns to provide QoS Routing support, and integration with ns mpls implementation. This project is hosted on sourceforge, and has its own mailing list and publically accessible cvs repository. Work on this would continue further and would include implementation/integration of other technologies like OSPF-TE and RSVP. By implementing this, we have succeeded in provided an environment in which network simulations based on these technologies can be performed. This would be a basis for much of the research work in these areas being carried out in Helsinki University of Technology under IRoNet research project (http://www.tct.hut.fi/tutkimus/ironet/index.shtml).

Although these technologies provide an impressive array of solutions for various need of current and future Internet, a lot of unanswered questions remain. For eg., although QoS Routing appears to be a very powerful concept, its utility is currently restricted to use for routing of large traffic aggregates such as traffic trunks which are part of one ISP. It cannot, in current form, be used as a normal routing replacement as doing QoS Routing for each packet as it arrives would be expensive and not too beneficial. What is needed is a class based service such as Diffserv which can group packets in behaviour aggregates, and use QoS Routing for traffic aggregates. This can greatly enhance the utility of Diffserv, while preserving low cost and simplic-

ity of Diffserv architecture, which is its main strength. As another example, it could be beneficial to augment current Qospf algorithm to include more heuristics such as support for jitter constraints to provide better paths. The utility of Qosrns in testing of such ideas is quite evident.

Qosrns is an ongoing work, and other students/researchers from HUT and IIT Guwahati would be joining the project as developers. Eventually, it would provide integrated support for all current QoS and Traffic Engineering technologies.

References

[A+01] Daniel O. Awduche et al., *RSVP-TE: Extensions to RSVP for LSP Tunnels*, Work in progress, August 2001.

[AC] Gaeil Ahn and Woojik Chun, *Mpls network simulator*, Tech. report, Chungnam National University, Korea.

[ACE+] Daniel O. Awduche, Angela Chiu, Anwar Elwalid, Indra Widjaja, and XiPeng Xiao, *Overview and Principles of Internet Traffic Engineering* , Work in progress.

[ADF+01] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, *LDP Specification*, RFC 3036, January 2001.

[AM+01] O. Aboul-Magd et al., *Constraint-Based LSP Setup using LDP*, Work in progress, February 2001.

[AMA+99] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, *Requirements for Traffic Engineering Over MPLS*, RFC 2702, September 1999.

[AWK+99] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda, *QoS Routing Mechanisms and OSPF Extensions*, RFC 2676, August 1999.

[BBC+98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *An Architecture for Differentiated Services* , RFC 2475, December 1998.

[BZB+97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Spec*, RFC 2205, September 1997.

[C+] Anna Charny et al., *An Expedited Forwarding PHB*, Work in progress.

[CNS98] E. Crawley, R. Nair, and B. Rajagopalan H. Sandick, *A Framework for QoS-based Routing in the Internet*, RFC 2386, August 1998.

[Fau01] F. Faucheur, *Mpls support of differentiated services*, Internet draft, draft-ietf-mpls-diff-ext-08.txt, IETF, February 2001.

[Gro] Dan Grossman, *New Terminology for Diffserv* , Work in progress.

[JA+02] B. Jamoussi, L. Andersson, et al., *Constraint based lsp setup using ldp*, Work in Progress 3212, January 2002.

[LN+98] T. Li, Juniper Networks, et al., *A provider architecture for differentiated services and traffic engineering*, RFC 2430, October 1998.

[LR] Raymond Law and Srihari Raghavan, *Diffserv and mpls - concepts and simulations*, Tech. report, Virginia Polytechnic Institute.

[Res] Researchers, *The ns manual*, Tech. report, U.C Berkeley, LBL, USC/ISI, and Xerox PARC, http://www.isi.edu/nsnam/ns/ns-documentation, The VINT project.

[RVC01] E. Rosen, A. Viswanathan, and R. Callon, *Multiprotocol Label Switching Architecture*, RFC 3031, January 2001.

[SJ02] P. Srisuresh and P. Joseph, *The lsas to extend ospf for traffic engineering*, Internet draft, IETF, 2002.

[SNH] N. Seddigh, B. Nandy, and J. Heinanen, *An Assured Rate Per-Domain Behaviour for Differentiated Services* , Work in progress.